# *Sistemas de Operação*

DETI

*Sistemas de ficheiros (Princípios)*

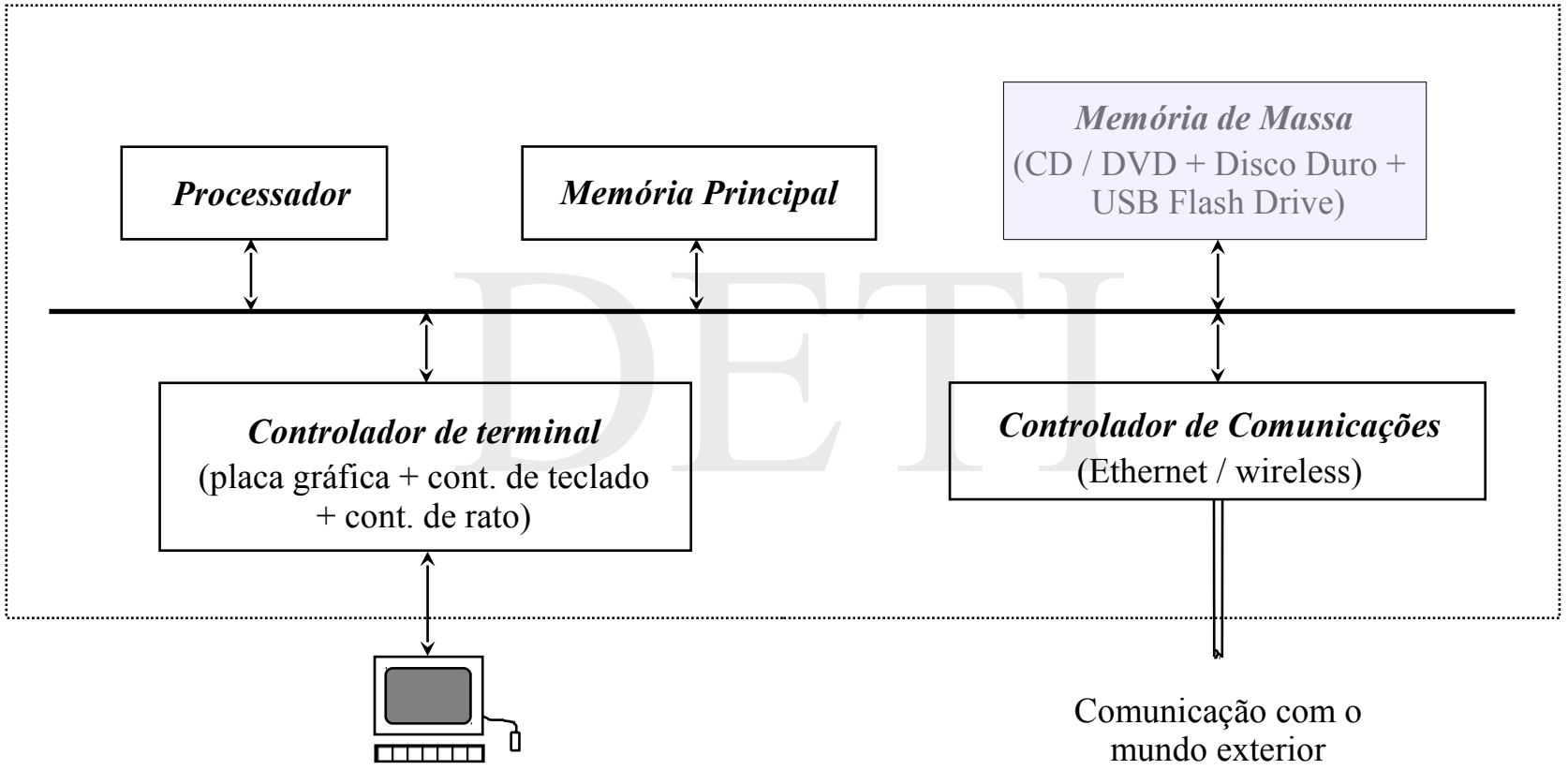
António Rui Borges / Artur Pereira

# *Sumário*

- *Memória de massa*
  - *Importância*
  - *Requisitos*
  - *Tipos*
  - *Abstração operacional*
- *Conceito de ficheiro*
  - *Atributos*
  - *Tipos de ficheiro*
  - *Operações*
- *Acesso a um ficheiro*
- *Implementação de um sistema de ficheiros*
  - *Gestão do espaço*
    - *Armazenamento contínuo*
    - *Armazenamento disperso*
      *listas ligadas, tabelas de referência única e múltiplas, tabela bitmap*
  - *Catalogação de objetos*
    - *Formato das entradas de directório*
      *atributos e nomes*
- *Disco magnético*
  - *Características*
  - *Cache de disco*
    - *Protocolo de acesso*
- *Leituras sugeridas*

DETI

# *Diagrama de blocos do sistema computacional*




DETI

# *Importância da memória de massa*

- Para que o sistema computacional realize trabalho útil, é necessário existir um local para armazenamento mais ou menos permanente das diferentes aplicações que são executadas. Este local é a *memória de massa (secundária)*.
- Quando um sistema computacional é ligado, só existe em *memória principal* (numa pequena região de tipo ROM), um programa, o *boot loader*, cuja função principal é ler de uma região específica da *memória de massa* um programa de maior dimensão que carrega em memória principal, e põe em execução, o programa que implementa o ambiente de interacção com o utilizador.
- Além disso, quase todas os programas, durante a sua execução, produzem, consultam e/ou alteram quantidades variáveis de informação que está armazenada de um modo mais ou menos permanente em *memória de massa*.

DEI

# *Requisitos da memória de massa*

- Um conjunto variado de requisitos são impostos à memória de massa
  - *ser não volátil* - a informação deve poder existir antes da criação do(s) processo(s) que a vai(ão) usar, e sobreviver à sua terminação, mesmo quando o sistema computacional é desligado
  - *ter grande capacidade de armazenamento* - a informação manipulada pelos processos pode ultrapassar em muito aquela que é diretamente armazenada nos seus espaços de endereçamento próprios
  - *providenciar um acesso eficiente* - o acesso à informação armazenada deve ser efetuado da maneira mais simples e rápida possível
  - *manter a integridade da informação* - a informação armazenada deve estar protegida contra a alteração e a corrupção acidentais
  - *permitir a partilha da informação* - a informação deve ser acessível concorrentemente a múltiplos processos que fazem uso dela

DETI

# *Tipos de memória de massa*

| **Tipo** | **Tecnologia** | **Capacidade** (Gbytes) | **Uso operacional** | **Velocidade de transferência** (Mbytes/s) |
|---|---|---|---|---|
| CD-ROM | mecânico / óptica | 0,7 | read | 0,5 |
| DVD | mecânico / óptica | 4 – 8 | read | 0,7 |
| HDD | mecânico / magnética | 250 – 4000 | read /write | 480 |
| USB FLASH | semicondutora | 2 – 256 | read /write | 30 (r) / 15 (w) |
| SDD | semicondutora | 64 – 512 | read /write | 500 |

# *Abstração operacional da memória de massa*

- A memória de massa pode ser percebida em termos de operacionais através de uma abstração muito simples
  - cada dispositivo é entendido como representando um *array* de NTBK blocos de armazenamento, cada um deles formado por BKSZ bytes (tipicamente, BKSZ varia entre 256 e 32K);
  - o acesso a cada bloco, para leitura ou escrita, é feito de um modo aleatório e o tempo associado não depende do tipo de operação.


DETI

# *Caracterização do problema*

- A manipulação direta da informação contida no dispositivo físico não pode ser deixada inteiramente à responsabilidade do programador de aplicações
- A complexidade inerente à sua estruturação e a necessidade de imposição de critérios de qualidade, relacionados com a eficiência no acesso, a integridade e a partilha, exigem a criação de um modelo uniforme de interação

DEI

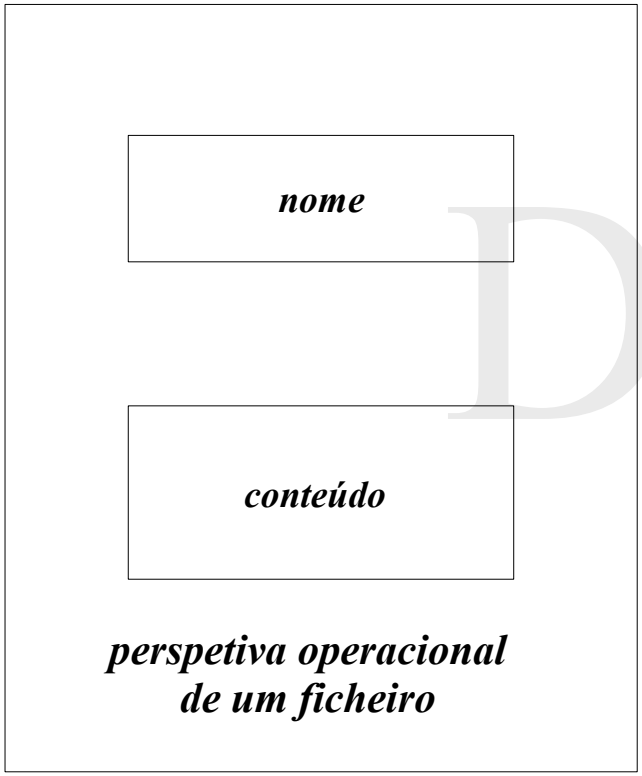

***dispositivo físico entendido como um array de blocos***

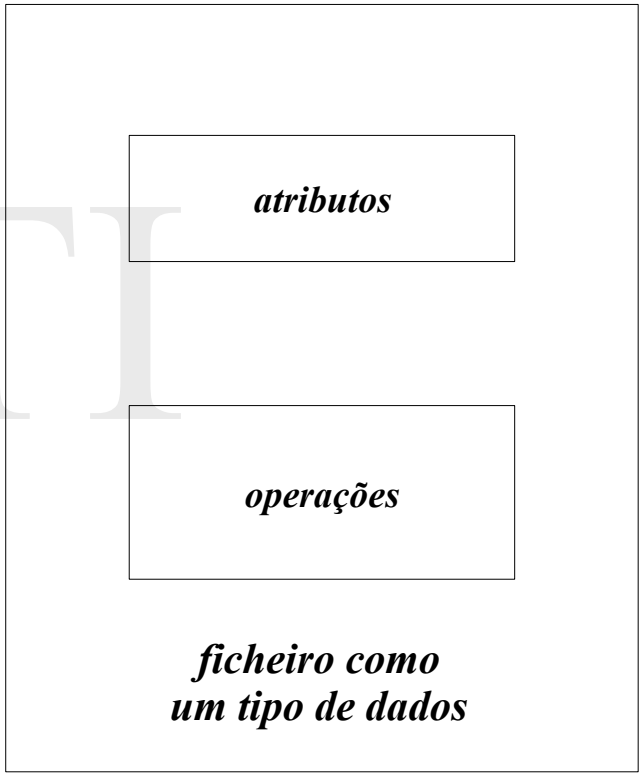
# *Conceito de ficheiro - 1*

- O conceito de *ficheiro* surge, assim, como a *unidade lógica de armazenamento em memória de massa*
- Pretende-se com isto dizer que a leitura e a escrita de informação se faz sempre no âmbito estrito de um *ficheiro*
- Um *ficheiro* apresenta como elementos básicos
  - *o nome* – a forma genérica de referenciar a informação
  - *o conteúdo* – a informação propriamente dita, organizada numa sequência de bits, bytes, linhas ou registos, cujo formato preciso é definido pelo criador do ficheiro e que tem que ser conhecido por quem a ele acede
- Sob o ponto de vista do programador de aplicações, um ficheiro é entendido como um tipo de dados abstrato, caracterizado por um conjunto de *atributos* e por um conjunto de *operações*

DEI

# *Conceito de ficheiro - 2*



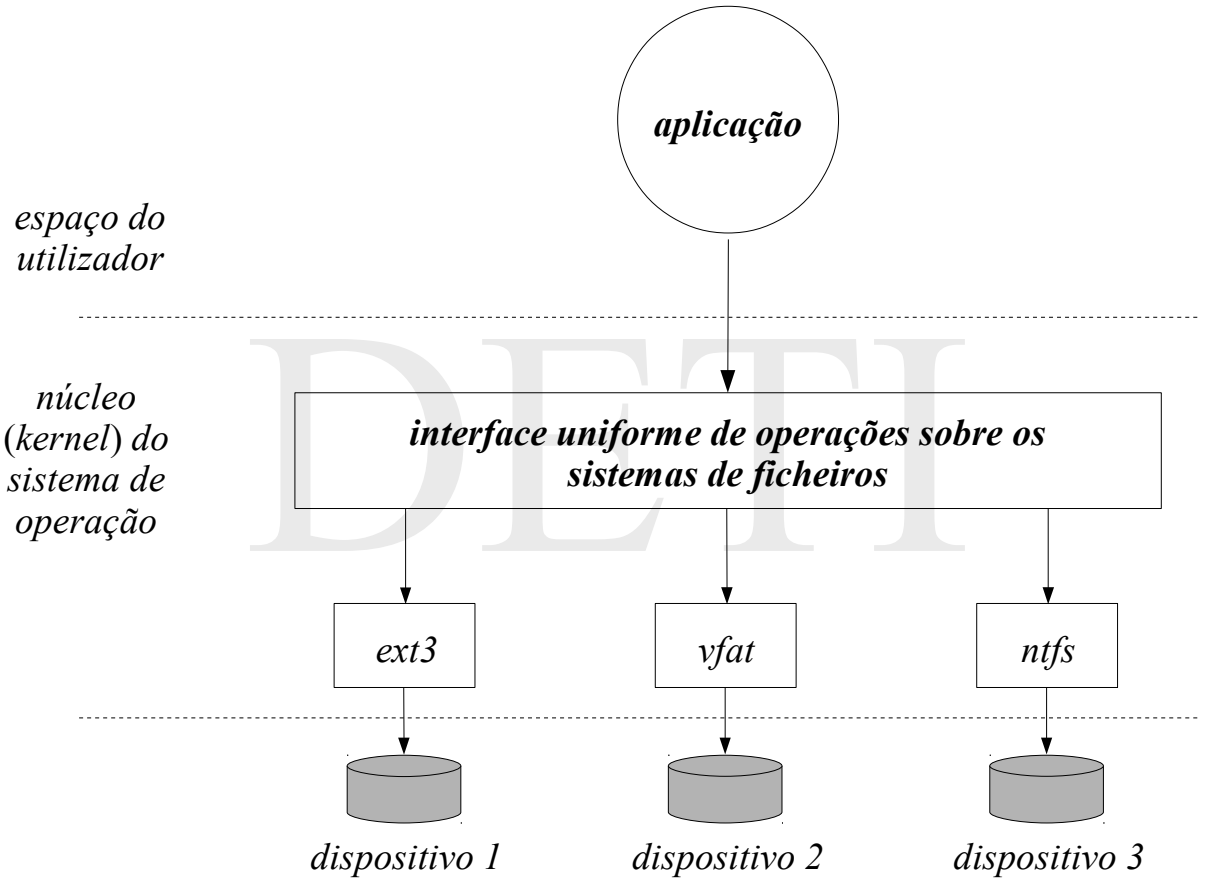

DETI

# *Conceito de ficheiro - 3*

- O papel do sistema de operação é implementar este tipo de dados, fornecendo um leque alargado de operações, *chamadas ao sistema*, que estabelecem um interface simples e seguro de comunicação com a memória de massa para a sua manipulação
- A parte do sistema de operação que se dedica a esta tarefa, designa-se por *gestor de sistemas de ficheiros*
- Diferentes implementações do tipo de dados *ficheiro* conduzem a diferentes tipos de *sistemas de ficheiros*
- Hoje em dia, os sistemas de operação implementam diferentes sistemas de ficheiros associados quer com diferentes dispositivos físicos, quer com o mesmo
  - Este aspecto facilita a *interoperacionalidade*, estabelecendo um meio comum de partilha de informação entre sistemas computacionais heterogéneos.

DETI

# *Conceito de ficheiro - 4*




DETI

# *Atributos de um ficheiro - 1*

- Os *atributos* de um ficheiro são variados e dependem da implementação específica. Destaca-se aqui um conjunto mínimo que está habitualmente presente
  - *nome* – o ficheiro passa a ter uma identidade própria com a sua nomeação, tornando-se independente do processo e do utilizador que o criaram e mesmo do sistema de operação em que foi criado
    - o nome é tipicamente formado por uma sequência de caracteres alfanuméricos e de outros caracteres com representação gráfica
    - é relativamente frequente que o *nome* de um ficheiro seja formado por duas partes, separadas por '.'

      <nome do ficheiro propriamente dito>.<extensão>

      a *extensão*, quando existe, procura indiciar de algum modo o tipo de organização interna apresentado pelos dados; num ambiente de interação gráfico, torna-se pos-sível associar uma aplicação com cada tipo de extensão, permitindo que, ao clicar o rato sobre o ícone representativo de um ficheiro particular, a aplicação associada seja posta em execução, leia o conteúdo do ficheiro e o apresente de algum modo ao utilizador

DETI

# *Atributos de um ficheiro - 2*

- *identificador* – o *nome*, enquanto elemento individualizador de um ficheiro, é adequado para uma referência externa (tipicamente de origem humana), mas é pouco prático em termos de organização interna; o *identificador* constitui, por isso, o elemento discriminador interno para acesso aos outros atributos do ficheiro e, através deles, à informação propriamente dita nele armazenada
- *tamanho* – o comprimento do conteúdo informativo, normalmente em n.º de bytes
- *pertença* – a indicação do utilizador que criou o ficheiro e a quem ele pertence
- *permissões* – o controlo de acesso ao ficheiro, especificando quem pode ler o seu conteúdo, escrever nele ou realizar operações de execução
- *monitorização de acesso* – datas e tempos dos instantes de criação, de último acesso e de última modificação
- *localização* – listagem ordenada dos blocos do dispositivo onde está armazenado o conteúdo informativo do ficheiro

DETI

# *Atributos de um ficheiro - 3*

- *tipo* – os sistemas de ficheiros admitem ficheiros de tipos diversos
  - *ficheiros regulares* – são os ficheiros convencionais, contêm todo o tipo de informação que é habitualmente armazenada em memória de massa
  - *directórios* – são ficheiros internos, com formato pré-definido, que descrevem a estrutura subjacente
  - *atalhos* – são ficheiros internos, com formato pré-definido, que descrevem a localização de um outro ficheiro através da especificação do seu encaminhamento na estrutura hierárquica pré-existente
  - *ficheiros especiais* – alguns sistemas operação fornecem uma visão integrada das operações de entrada e saída de dados, usando o mesmo conjunto de operações, quer para acesso a ficheiros, quer para acesso a dispositivos genéricos de entrada-saída; quando existem, os *ficheiros especiais* modelam estes dispositivos e dividem-se em ficheiros especiais de *tipo carácter* ou de *tipo bloco*, conforme a transferência possa ocorrer byte a byte, ou apenas através de um grupo de bytes de comprimento pré-definido

DEI

# *Diretórios - 1*

- Os *diretórios* servem para elencar todos os ficheiros que são visíveis a um dado nível da organização hierárquica interna, facilitando, portanto, o acesso a um ficheiro particular.
- O seu conteúdo pode ser entendido como um *array* [de tamanho variável] de *entradas de diretório*, em que cada *entrada* descreve o *nome* de um ficheiro e, direta ou indiretamente, referencia os seus *atributos*.

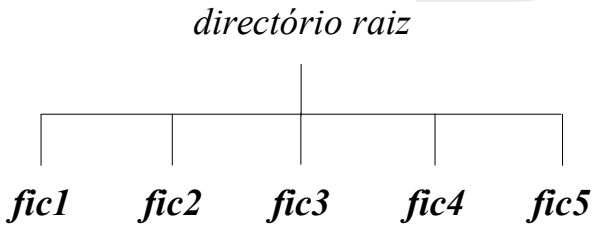
DETI

| | ***nome*** | ***referência direta ou indireta aos atributos*** |
|---|---|---|
| *ent*[0] | | |
| *ent*[1] | | |
| *ent*[2] | | |
| • | | |
| • | | |
| • | | |
| *ent*[N-1] | | |

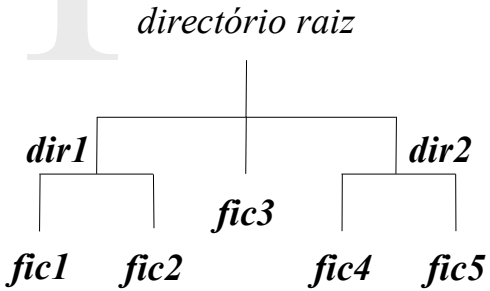
# *Diretórios - 2*

- Os dispositivos físicos atuais, face à grande capacidade de armazenamento interno, podem conter em condições normais milhões de ficheiros regulares. Se todos eles estivessem colocados ao mesmo nível, a referência a um deles em particular, ou a atribuição de um nome a um novo ficheiro, tornar-se-iam operações quase impossíveis de realizar na prática pela complexidade que lhes está inerente.
- Assim, qualquer sistema de ficheiros contemporâneo pressupõe uma *organização hierárquica da informação*.

DETI




***sistema de directório único***




***sistema hierárquico de directórios***
**(*árvore de directórios*)**

O *diretório raiz* denota a origem do sistema de ficheiros.

# *Diretórios - 3*

- A introdução de uma hierarquia de directórios num sistema de ficheiros faz com que o *nome* de um ficheiro deixe de constituir por si só um elemento individualizador absoluto
- Não só em diferentes diretórios passam a poder existir ficheiros com o mesmo *nome*, como também, dado apenas o *nome* do ficheiro, deixa de ser possível localizá-lo de uma maneira eficiente
- Para o conseguir, torna-se necessário incluir na sua nomeação, como prefixo, o caminho a percorrer na hierarquia de diretórios desde um local pré-estabelecido até ao local de referência do ficheiro
- Isto pode ser realizado de duas maneiras
  - *encaminhamento absoluto* – o ponto de partida é o diretório raiz
  - *encaminhamento relativo* – o ponto de partida é o chamado *diretório de trabalho*; o directório onde a aplicação que referencia o ficheiro é lançada, ou que é especificado como tal durante a execução da aplicação.

DETI

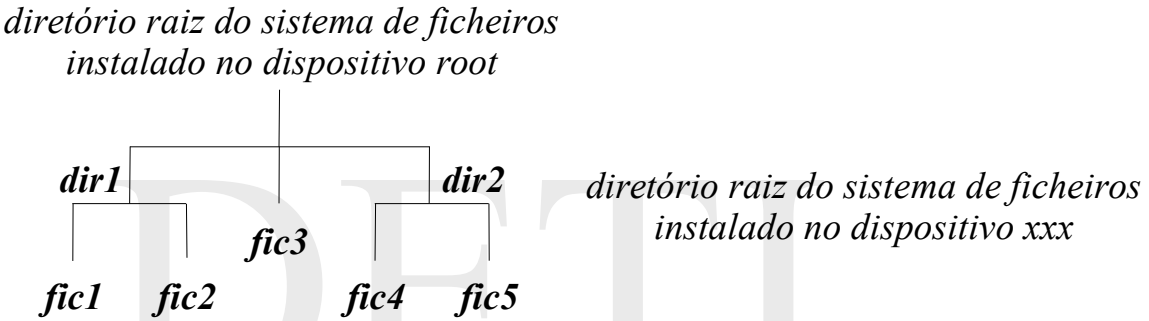
# *Diretórios - 4*

- A especificação do caminho é efetuada listando o *nome* dos diretórios que são sucessivamente atravessados até se atingir o ficheiro que é nomeado. O caracter usado como elemento separador é '/' em Unix e '\' em Windows
- Há dois nomes reservados que têm tradicionalmente um significado pré-estabelecido: o nome "." que representa o diretório de trabalho, e o nome ".." que representa o diretório imediatamente acima do atual na hierarquia de diretórios do sistema de ficheiros
- Em Unix, existe uma única hierarquia de diretórios que integra todos os sistemas de ficheiros residentes nos vários dispositivos de memória de massa presentes no, ou acessíveis pelo, sistema computacional. O topo da hierarquia, denotado por '/', corresponde ao directório raiz do sistema de ficheiros onde está armazenado o código do sistema de operação (dispositivo *root*). Os sistemas de ficheiros restantes são *montados* em diretórios particulares deste sistema de ficheiros
- Em Windows, cada dispositivo de memória de massa presente no, ou acessível pelo, sistema computacional tem a sua própria hierarquia e é identificado por uma letra do alfabeto seguida de ':' (`A:`, `B:`, etc.)

DETI

# *Diretórios - 5*

## *Unix*




DETI

**encaminhamento absoluto**

`/` (nomeação do diretório raiz)

`/dir1/fic1` (nomeação do ficheiro fic1)

`/dir2` (nomeação do diretório dir2)

**encaminhamento relativo**

(assumindo `/dir1` como o diretório de trabalho)

`..` (nomeação do diretório raiz)

`../dir2` (nomeação do diretório dir2)

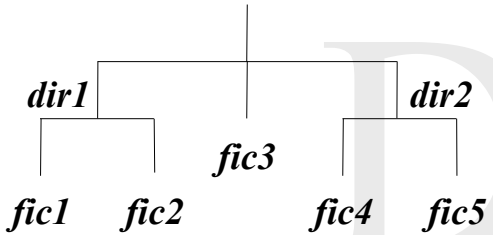
`fic2` (nomeação do ficheiro fic2)

`./fic2` (nomeação do ficheiro fic2)

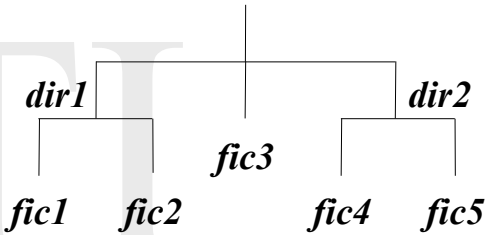
# Diretórios - 6

## Windows

*diretório raiz do sistema de ficheiros instalado no dispositivo* `C:`




*diretório raiz do sistema de ficheiros instalado no dispositivo D:*




DETI

**encaminhamento absoluto**

`C:\` (nomeação do diretório raiz do dispositivo C:)

`C:\dir1\fic1` (nomeação do ficheiro fic1 de C:)

`D:\dir2` (nomeação do diretório dir2 de D:)

**encaminhamento relativo**

(assumindo `C:\dir1` como o diretório de trabalho)

`..` (nomeação do directório raiz do dispositivo C:)

`..\dir2` (nomeação do diretório dir2 de C:)

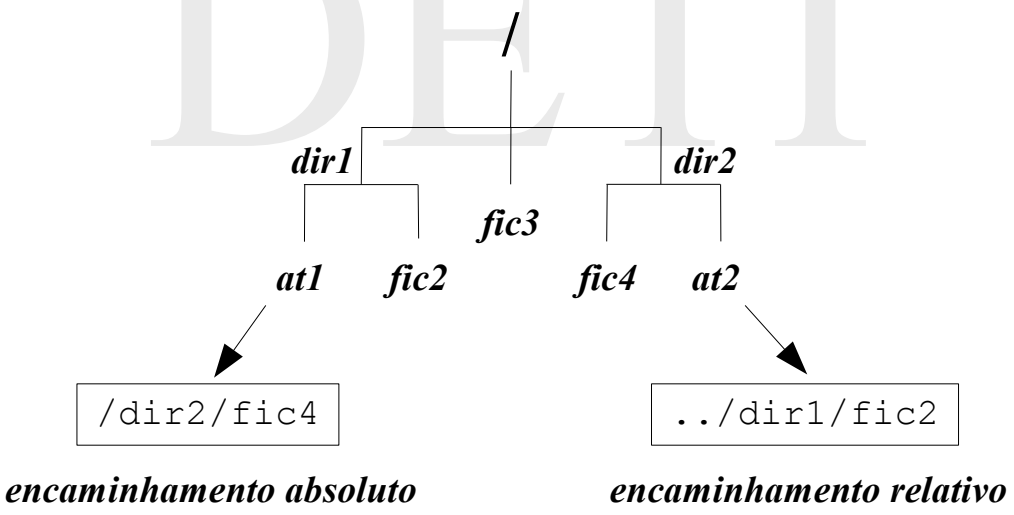
`fic2` (nomeação do ficheiro fic2 de C:)

`.\fic2` (nomeação do ficheiro fic2 de C:)

# *Atalhos - 1*

- Os *atalhos* servem para localizar um outro ficheiro, tipicamente um *ficheiro regular* ou um *directório*, através da especificação do seu encaminhamento na estrutura hierárquica pré-existente.

DETI



# *Atalhos - 2*

- Razões que levam à criação de *atalhos:*
  - *transparência de acesso a sucessivas versões da mesma aplicação* – desde que o acesso a uma aplicação seja efetuado sempre através de um atalho, quando uma nova versão é instalada, é apenas necessário alterar o atalho e fazê-lo referenciar a nova versão
  - *redirecionar o acesso a zonas de dados protegidas* – algumas aplicações de sistema mantêm zonas de dados de acesso restrito, mas permitem que os utilizadores normais criem neste contexto as suas próprias zonas de dados; através da criação de atalhos para diretórios nas zonas de cada um, torna-se possível permitir aos utilizadores individuais que criem os seus próprios dados sem comprometer a segurança geral do sistema
  - *acesso a diretórios e a ficheiros regulares residentes em sistemas de ficheiros distintos* – quando o sistema de operação impõe que cada dispositivo de memória de massa tenha a sua própria hierarquia de directórios, a criação de atalhos possibilita a implementação de um acesso cruzado
- Nada impede em princípio que o conteúdo de um *atalho* referencie um outro *atalho*, criando-se um cadeia de *atalhos*. A única coisa que tem que ser garantida é que se trata de uma cadeia linear. ***Porquê*?**

DEI

# *Operações sobre um ficheiro regular - 1*

- As *operações* que se podem realizar sobre um *ficheiro regular* são também variadas e dependem do sistema de operação. Há, porém, um núcleo base que de algum modo está sempre presente
  - *criação* – é reservada no directório uma entrada para o ficheiro regular; a entrada vai conter o *nome* e os atributos, ou uma referência para os atributos, que são inicializados, nalguns casos usam-se valores passados como parâmetros da operação; o *tamanho* do ficheiro, em particular, é colocado a zero, sinalizando que está vazio e admitindo, por isso, só operações de *escrita*
  - *abertura* – é a operação alternativa à *criação*, sendo efetuada sempre que o ficheiro referenciado já existe; as entradas do directório são pesquisadas para localizar a entrada respetiva e para aceder aos atributos
    - as operações de *criação* e de *abertura* efetuam-se sempre que se pretende estabelecer comunicação com um ficheiro regular; o objectivo é construir em memória principal uma estrutura de dados que permita nas operações subse-quentes um acesso mais eficiente à informação

DEI

# *Operações sobre um ficheiro regular - 2*

- *fecho* – traduz o encerramento da comunicação com o ficheiro regular; o bloco de dados relativo à parte do conteúdo que estava a ser referenciado na altura, é escrito no dispositivo, se tiver ocorrido alteração; os atributos relevantes são atualizados com a informação da estrutura de dados residente em memória principal e o espaço ocupado por ela é libertado
- *posicionamento* – o conteúdo do ficheiro regular é percebido pelo programador de aplicações como um *continuum* de dados; o local nesse *continuum* onde irá decorrer a próxima transferência de informação pode ser definido através da operação de posicionamento que fixa a chamada *posição atual*
  - quando um ficheiro regular é *criado* ou *aberto*, a *posição actual* aponta para o início desse *continuum*; a realização subsequente de operações de *escrita* ou de *leitura* conduzem à atualização do valor de *posição actual*, que passa a apontar para a posição imediatamente a seguir à localização de escrita ou de leitura do último dado

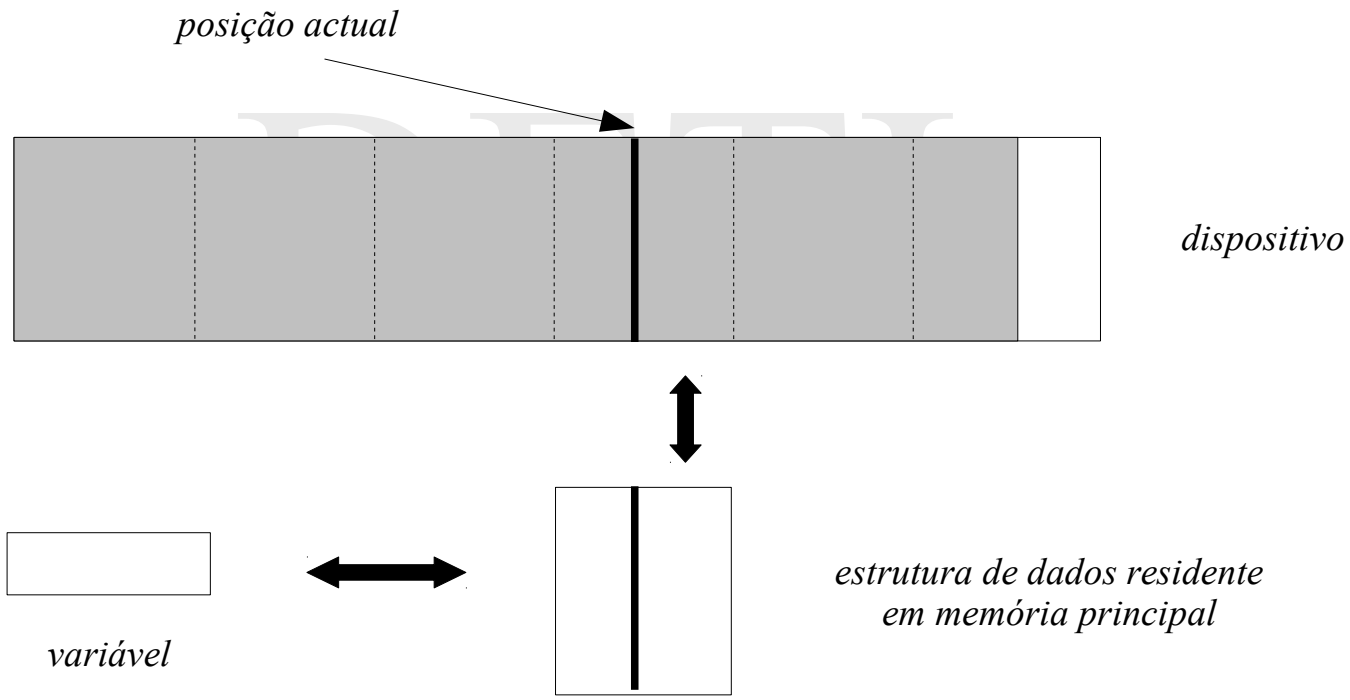
DETI

# *Operações sobre um ficheiro regular - 3*

- *escrita* – é a operação de transferência de informação *para o* ficheiro regular; os dados são escritos a partir da *posição atual* e retirados de uma variável cuja localização é passada como parâmetro da operação
- *leitura* – é a operação de transferência de informação *do* ficheiro regular; os dados são lidos a partir da posição atual e armazenados numa variável cuja localização é passada como parâmetro da operação
  - a transposição da abstração do *continuum* para o armazenamento efetivo dos dados em blocos do dispositivo físico impõe a tradução da *posição atual* num bloco específico e num deslocamento dentro desse bloco;
  - assim, a realização das operações de *escrita* e de *leitura* exigem a monitorização constante do bloco onde são efetuadas: quando se iniciam, é necessário garantir que o bloco foi transferido para a estrutura de dados residente em memória principal; depois, durante a execução das operações, se os limites do bloco forem ultrapassados, é necessário escrever o bloco em causa no dispositivo, caso tenha ocorrido uma alteração, e ler do dispositivo o bloco seguinte

# *Operações sobre um ficheiro regular - 4*

***conteúdo do ficheiro visto como um continuum de dados***



# *Operações sobre um ficheiro regular - 5*

- *apagamento* – o ficheiro é removido do sistema de ficheiros; os blocos de dados de armazenamento do seu conteúdo informativo são libertados e a entrada do directório correspondente, bem como a região onde estão armazenados os restantes atributos, se não estiverem localizados na entrada de directório, são sinalizadas vazias.

DEEI

# *Operações sobre um diretório - 1*

- As *operações* que se podem realizar sobre um *diretório* são também variadas e dependem do sistema de operação. Há, porém, um núcleo base que de algum modo está sempre presente
  - *criação* – é reservada no diretório uma entrada para o novo diretório; a entrada vai conter o *nome* e os atributos, ou uma referência para os atributos, que são inicializados, nalguns casos usam-se valores passados como parâmetros da operação; ao contrário do que se passa com um *ficheiro regular*, o *tamanho* do diretório não é colocado a zero, um diretório vazio contém sempre referências a si próprio e ao diretório imediatamente acima na hierarquia para que a navegação na *árvore* de diretórios possa ser implementada de um modo eficiente
  - *apagamento* – o diretório é removido do sistema de ficheiros; tal como para um *ficheiro regular*, os blocos de dados de armazenamento do seu conteúdo são libertados e a entrada do diretório correspondente, bem como a região onde estão armazenados os restantes atributos, se não estiverem localizados na entrada de diretório, são sinalizadas vazias; o diretório tem que estar vazio para que a operação possa ter lugar

DET

# *Operações sobre um diretório - 2*

DETI

- *listagem do conteúdo* – as entradas de directório ocupadas são lidas para se obter uma descrição dos atributos dos ficheiros que são visíveis a esse nível da organização hierárquica interna.

# *Operações sobre um atalho*

- As *operações* que se podem realizar sobre um *atalho* são também variadas e dependem do sistema de operação. Há, porém, um núcleo base que de algum modo está sempre presente
  - *criação* – é reservada no directório uma entrada para o atalho; a entrada vai conter o *nome* e os atributos, ou uma referência para os atributos, que são inicializados, nalguns casos usam-se valores passados como parâmetros da operação; ao contrário do que se passa com um *ficheiro regular*, o *tamanho* do atalho é colocado num valor que corresponde ao tamanho do *string* que define o encaminhamento e o conteúdo informativo contém esse encaminhamento
  - *apagamento* – o atalho é removido do sistema de ficheiros; tal como para um *ficheiro regular*, os blocos de dados de armazenamento do seu conteúdo são libertados e a entrada do directório correspondente, bem como a região onde estão armazenados os restantes atributos, se não estiverem localizados na entrada de directório, são sinalizadas vazias
  - *listagem do conteúdo* – o conteúdo do atalho é lido para ser posteriormente usado como um caminho na localização de um outro ficheiro.

DEI

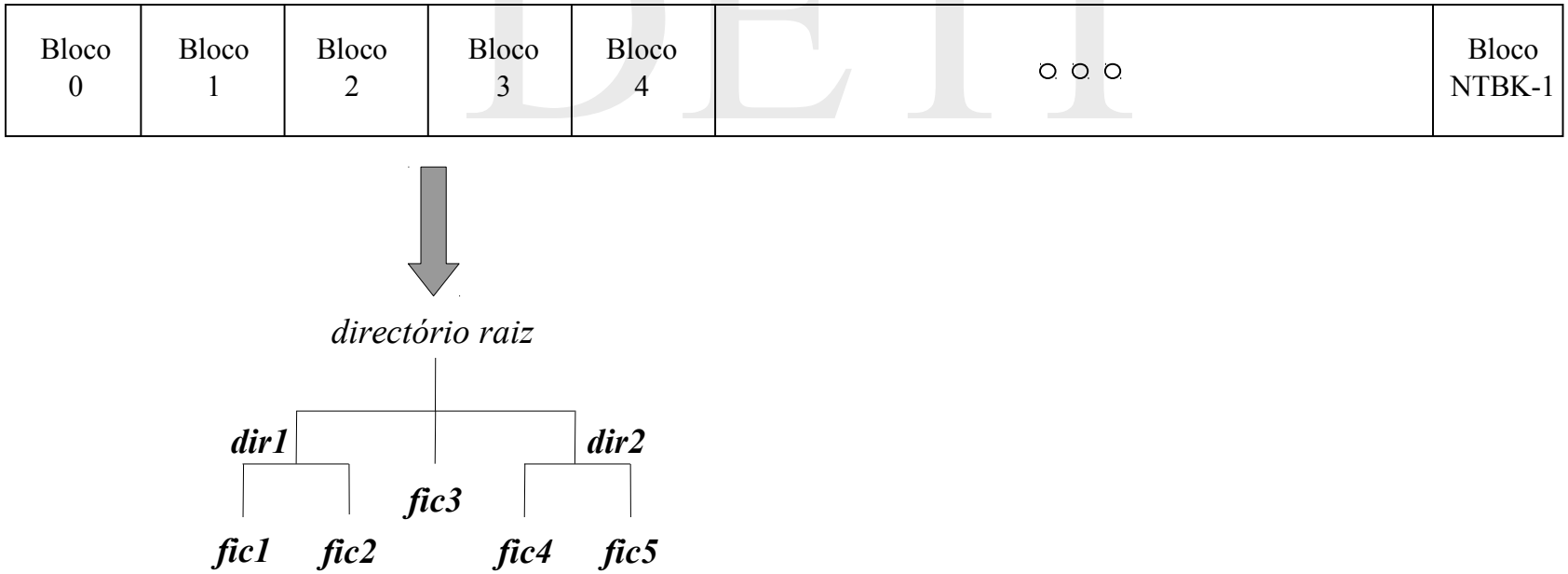
# *Operações sobre um ficheiro*

- *consulta dos atributos* – permite o acesso à generalidade dos atributos de um ficheiro, o que é importante em muitas situações concretas;
- *alteração dos atributos* – permite a modificação de alguns dos atributos de um ficheiro após sua criação, tais como a *pertença* e a *proteção*;
- *alteração do nome* – o nome do ficheiro é modificado
  - as duas últimas operações, bem como a operação de *apagamento*, são tipicamente efectuadas apenas por um processo lançado pelo utilizador a quem o ficheiro atualmente pertence.

DEI

# *Implementação de um sistema de ficheiros - 1*

- O principal problema que se coloca ao programador de sistemas na implementação de um sistema de ficheiros é como organizar o espaço de armazenamento interno do dispositivo, entendido como um *array* de blocos, de modo a fornecer a visão abstracta esperada pelo programador de aplicações, aquilo que se designa de *arquitetura interna*.




DETI

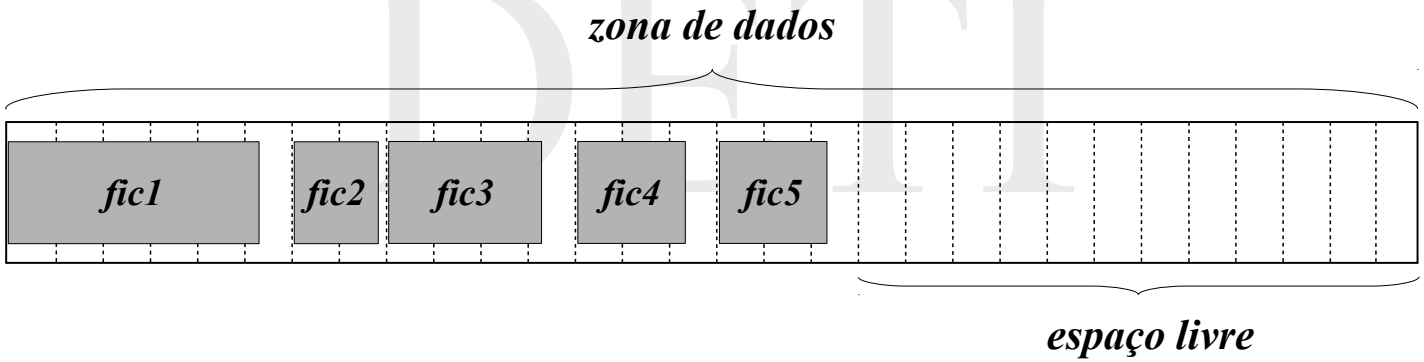
# *Implementação de um sistema de ficheiros - 2*

- Especificar uma *arquitetura interna* significa, em primeiro lugar, isolar no *array* de blocos diferentes regiões a que se atribui um significado bem definido e, em segundo lugar, conceber estruturas de dados que descrevam de uma forma precisa e adequada a funcionalidade prescrita (a *adequação* procura enfatizar a existência de algoritmos para a sua manipulação eficiente).
- Embora cada situação concreta seja diferente, pelo menos algumas das regiões seguintes costumam ser consideradas
  - *cabeçalho* (designado comummente de *superbloco*) – quando existe, contém elementos base que caracterizam o dispositivo e a arquitetura interna
  - *gestão do espaço* – descreve o estado de ocupação do dispositivo (que blocos do *array* incluídos na *zona de dados* estão ocupados e que blocos estão livres)
  - *catalogação dos objectos* – quando as *entradas de diretório* não armazenam diretamente os atributos do ficheiro que referenciam, mas contêm apenas uma referência à localização desses atributos, é necessário existir uma região onde essa informação seja armazenada
  - *zona de dados* – local onde estão especificamente armazenados os conteúdos informativos dos ficheiros.

# Gestão do espaço - 1

## Armazenamento contínuo

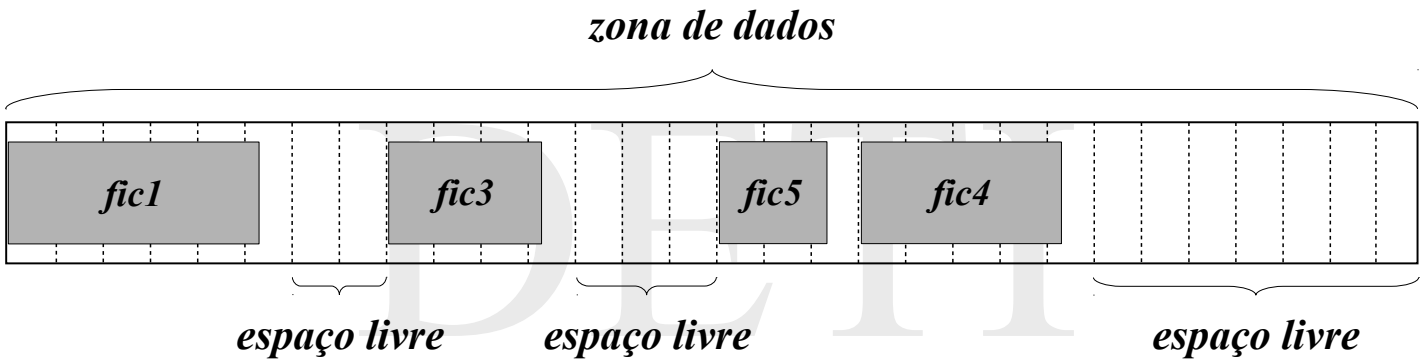
- A estratégia mais simples de gestão do espaço livre é proceder ao armazenamento sucessivo e contínuo do conteúdo informativo dos diferentes ficheiros na *zona de dados*.




- Há, neste caso, um aproveitamento quase total do espaço, ocorrendo apenas *fragmentação interna* (só parte do último bloco do conteúdo informativo de cada ficheiro é eventualmente desperdiçado). O espaço livre está concentrado no fim. Quer o espaço livre, quer o conteúdo informativo de cada ficheiro, pode ser descrito apenas por dois campos: *início* e *tamanho*.

# *Gestão do espaço - 2*

- O que acaba de ser dito é válido quando o sistema de ficheiros não permite a alteração do conteúdo dos ficheiros armazenados, nem o seu apagamento. Se tal for possível, vão surgir eventualmente 'buracos' no *continuum* de armazenamento.




- Aqui ocorre não só *fragmentação interna*, mas também *fragmentação externa* (o espaço livre deixa de estar compactado e surge disperso por múltiplas regiões). A sua gestão exige a formação de uma lista de regiões livres, cada uma descrita por um *início* e um *tamanho*, o que é muito pouco eficiente.
- Assim, esta organização é tradicionalmente restrita a sistemas de *backup*, residentes em bandas magnéticas ou discos de leitura óptica, ou a informação *read-only*, residente em discos de leitura óptica.

# *Gestão do espaço - 3*

## Armazenamento disperso

- Para eliminar o problema, pode considerar-se em alternativa ao armazenamento contínuo o armazenamento disperso. Nesta situação, quer o conteúdo informativo de cada ficheiro, quer o espaço livre, são descritos pelos agregados de blocos que lhes estão associados.

  Maneiras principais de o fazer

  - recorrendo a *listas ligadas em que os nós são os próprios blocos*
  - recorrendo a uma *tabela de referência* única
  - recorrendo a uma *tabela de referência bit-map* (apenas aplicável à gestão do espaço livre)
  - recorrendo a múltiplas *tabelas de referência*.

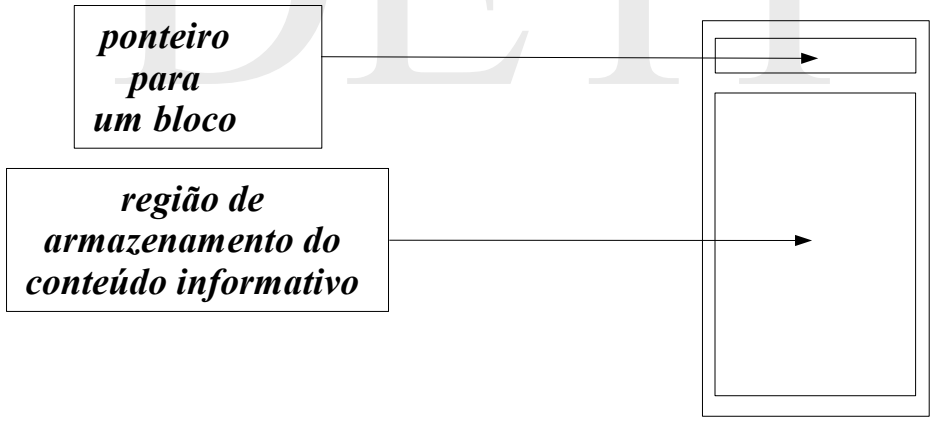
DETI

# Gestão do espaço - 4

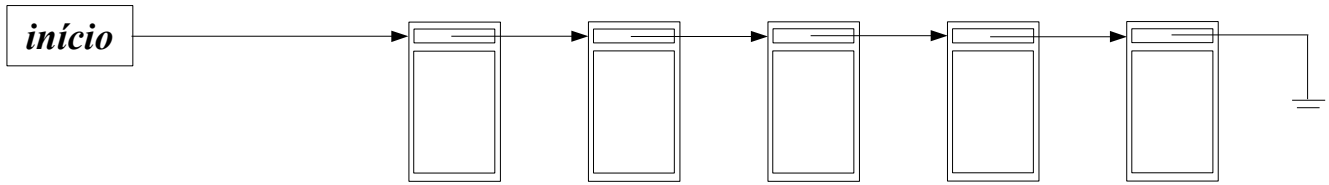
## *Listas ligadas em que os nós são os próprios blocos*

- Cada bloco da *zona de dados* é descrito por uma estrutura de dados de dois campos representativa de um nó de uma lista ligada a que ele estará associado: um *ponteiro para um bloco* e a *região de armazenamento do conteúdo informativo*.

DETI



# *Gestão do espaço - 5*




***lista ligada representativa do conteúdo informativo de um ficheiro específico ou do espaço livre***

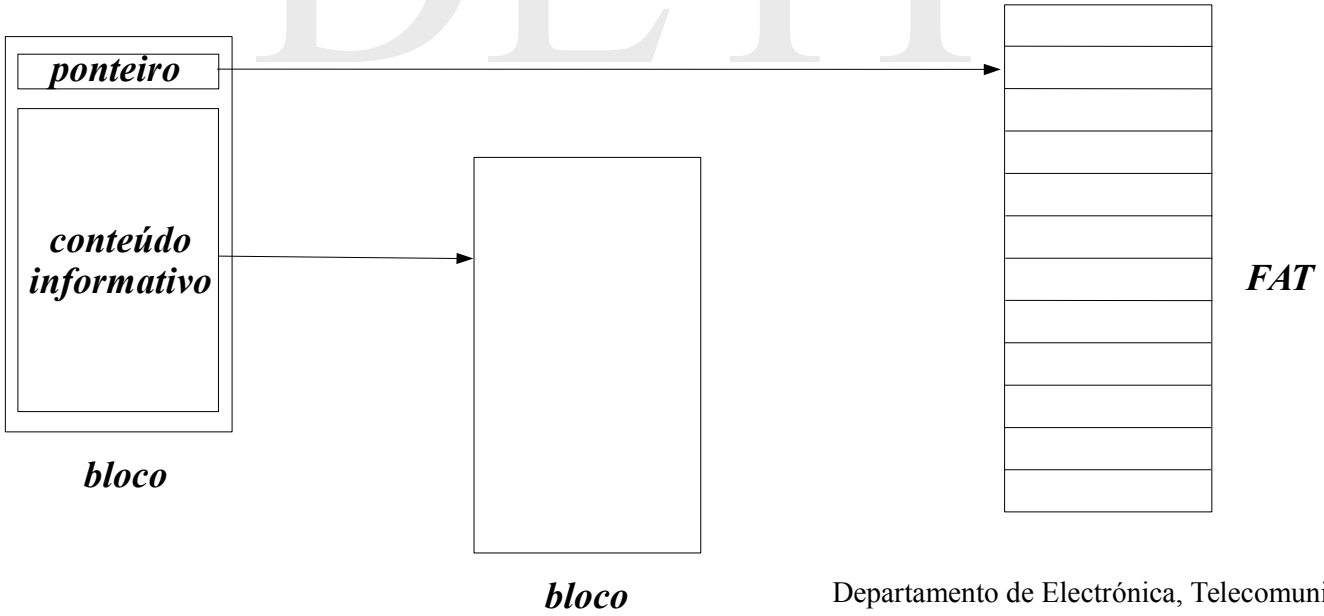
DETI

- Tal como para a situação de armazenamento contínuo, quer o espaço livre, quer o conteúdo informativo de um ficheiro, pode ser descrito apenas por dois campos: *início* e *tamanho*. Adicionalmente, como se pode constatar, a *fragmentação externa* é completamente eliminada em todas as situações (criação, apagamento de um ficheiro ou alteração do seu tamanho).
- Tem, porém, uma desvantagem se aplicada à descrição dos blocos que formam o conteúdo informativo de um ficheiro: em *acesso aleatório*, a localização de um bloco particular exige sempre o percurso da lista desde o seu início! O que é manifestamente ineficiente, sobretudo porque é realizado através de leituras sucessivas de blocos do dispositivo.

# Gestão do espaço - 6

## *Tabela de referência única*

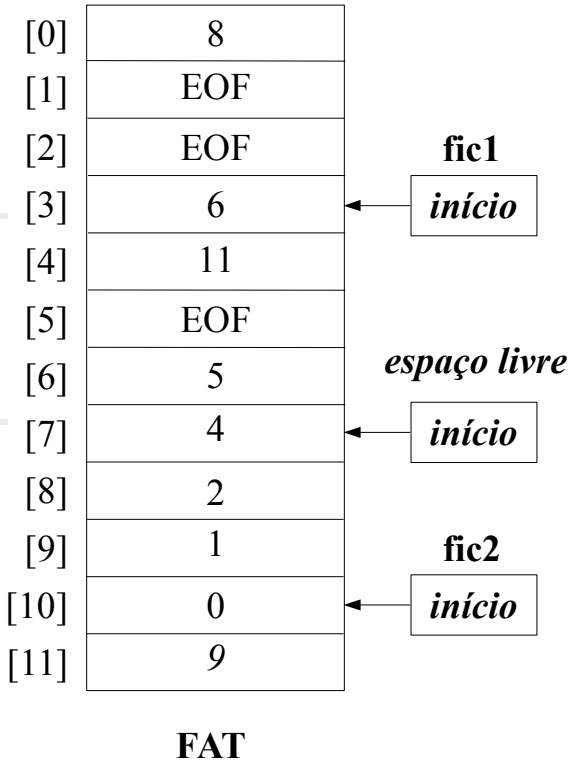
- Esta ineficiência pode ser atenuada em parte se se separar os dois campos que constituem a estrutura de dados que define um bloco. O *ponteiro* é colocado numa tabela especial, designada de FAT (*File Allocation Table*) e residente numa região pré-definida do *array* de blocos, que tem tantas entradas quantos os blocos da *zona de dados*. O bloco é reservado assim apenas para *armazenamento do conteúdo informativo*.




DETI

# *Gestão do espaço - 7*

- Tipicamente, o valor do *ponteiro* é o *número lógico do bloco*, isto é, o índice do elemento no *sub-array* de blocos que constitui a *zona de dados*.
- No exemplo ilustrado ao lado, são representados dois ficheiros: o ficheiro **fic1**, cujo conteúdo informativo está armazenado nos blocos 3, 6 e 5 e o ficheiro **fic2**, cujo conteúdo informativo está armazenado nos blocos 10, 0, 8 e 2. O espaço livre, por sua vez, está organizado numa lista formada pelos blocos 7, 4, 11, 9 e 1.
- O número de leituras necessárias para o acesso aleatório a um ficheiro é muito reduzido, ou mesmo eliminado se a tabela puder ser mantida permanentemente em memória.




DEI

# *Gestão do espaço - 8*

## *Tabela de referência bit-map*

- A gestão do espaço livre pode ser ainda feita de uma maneira mais compacta se se usar uma *tabela de referência bit-map*, ou seja, uma tabela especial, igualmente residente numa região pré-definida do *array* de blocos, que, tal como a FAT, tem tantas entradas quantos os blocos da *zona de dados*. A diferença está que cada entrada aqui é formada por um único bit, assumindo o valor 1, se o bloco correspondente estiver livre, e 0, em caso contrário.
- Note-se que este tipo de descrição não contempla qualquer ordenação de blocos, não podendo por isso ser aplicada à descrição do conteúdo informativo de um ficheiro.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] | [10] | [11] |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |

***tabela de referência bit-map***

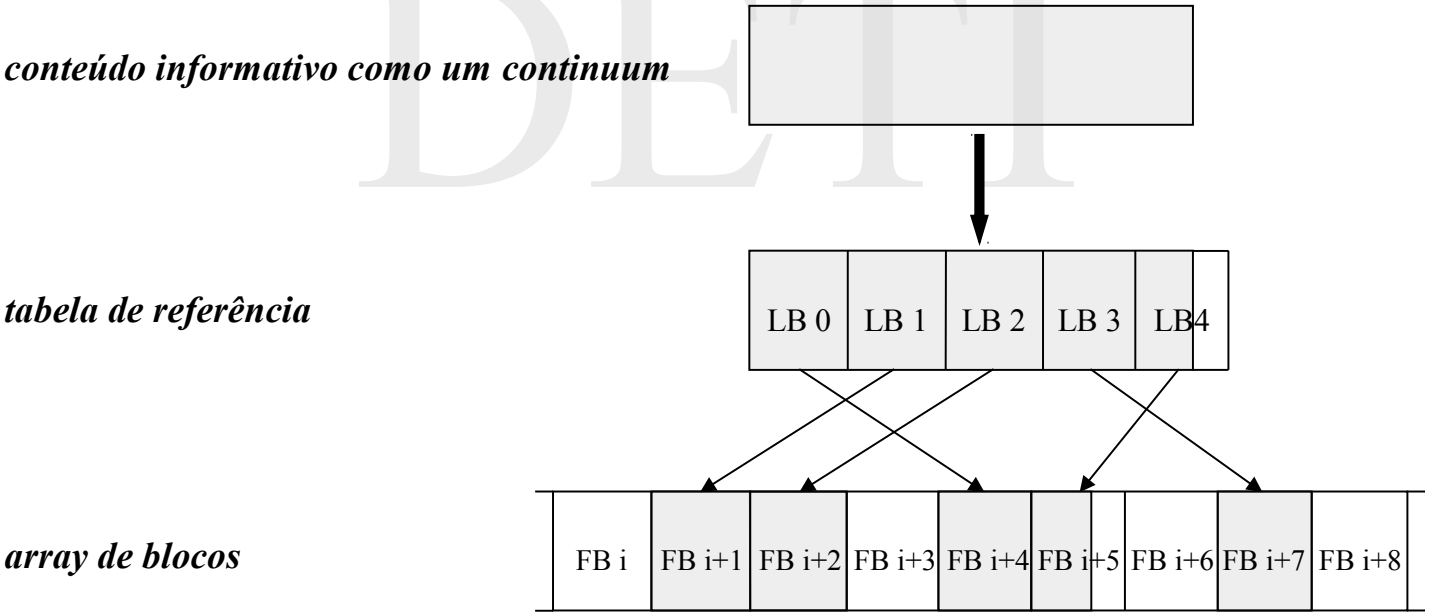
DETI

# Gestão do espaço - 9

## *Múltiplas tabelas de referência*

- O número de leituras a blocos do dispositivo, no caso de acesso aleatório a um ficheiro, é minimizado se os seus *atributos* contiverem a lista de referências ao conteúdo informativo.

DETI

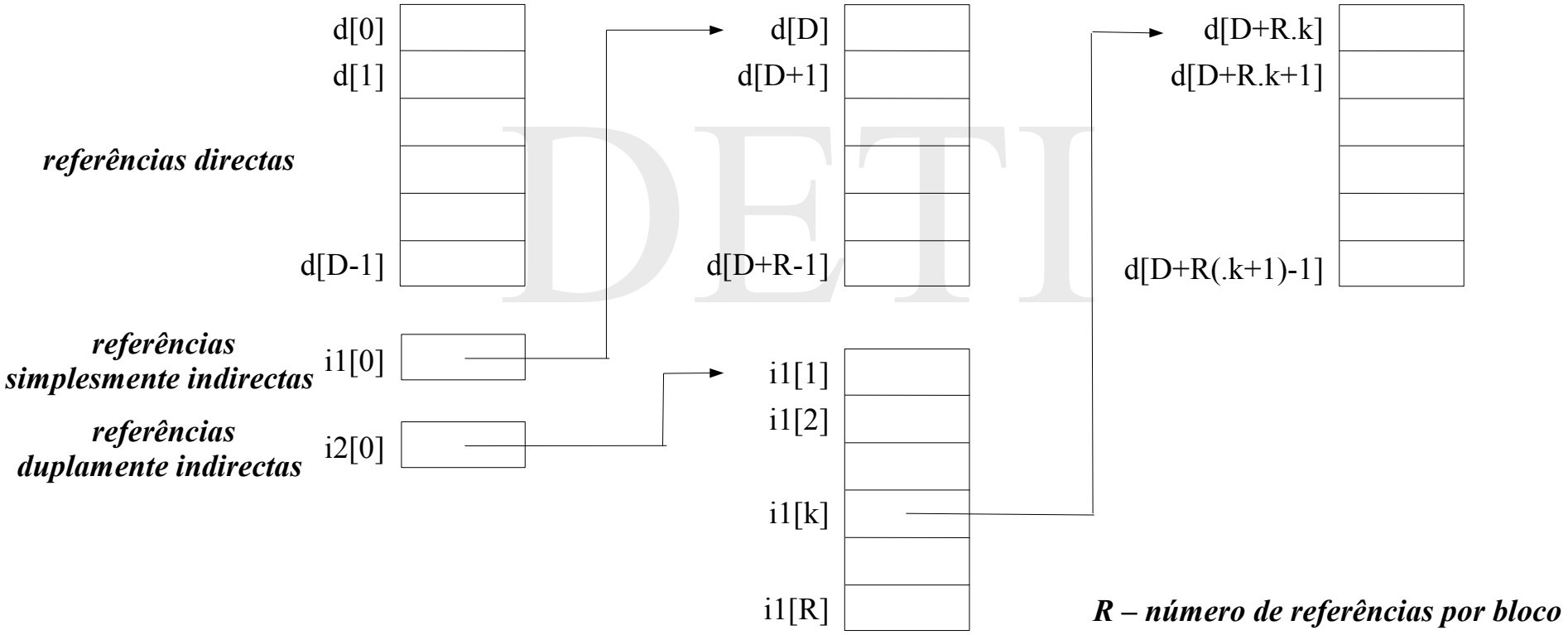


# *Gestão do espaço - 10*

- O problema que se coloca com este tipo de descrição é que, para incluir a tabela completa de referências ao conteúdo informativo de um ficheiro na estrutura de dados que define os seus *atributos*, seria necessário que essa estrutura tivesse um comprimento variável. Isto é inconveniente porque impediria o estabelecimento de uma estrutura regular ou, alternativamente, limitaria fortemente o tamanho máximo que um ficheiro poderia assumir.
- Assim, é costume organizar-se a tabela de forma hierárquica como um compromisso que otimiza o acesso aleatório ao conteúdo informativo de ficheiros 'pequenos' e simultaneamente permite a existência de ficheiros muito 'grandes'.

# Gestão do espaço - 11

## Organização hierárquica da tabela de referência




DETI

# *Gestão do espaço - 12*

- No exemplo ilustrado em que o contributo da localização para a estrutura de dados que define os *atributos* é de um array de D elementos e de duas variáveis:
  - ficheiros com tamanho até DxBKSZ bytes não necessitam de qualquer leitura intercalar de blocos para efetuarem um acesso aleatório a qualquer ponto do seu conteúdo informativo
  - ficheiros maiores mas com tamanho até (D+R)xBKSZ bytes necessitam no máximo de uma leitura intercalar
  - ficheiros maiores mas com tamanho até ($D+R+R^2$)xBKSZ bytes necessitam no máximo de duas leituras intercalares.

DET

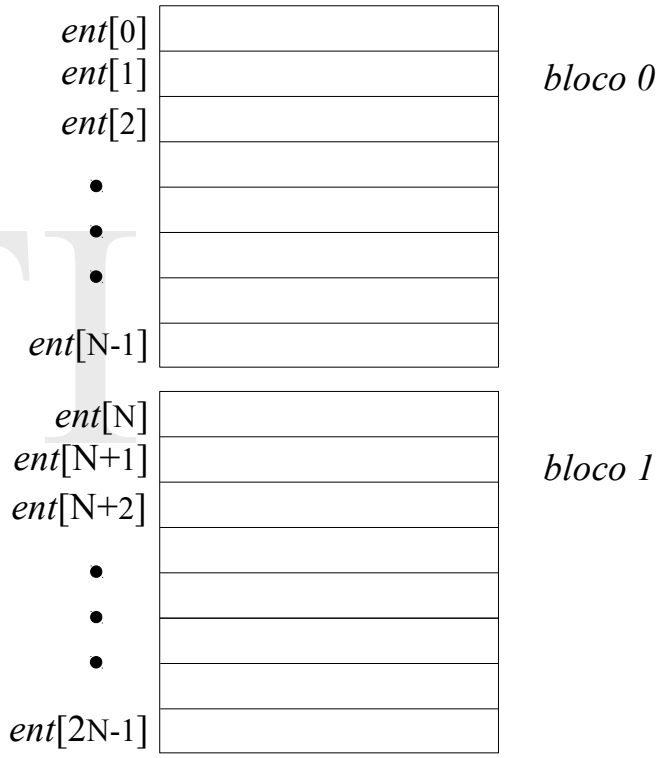
# *Catalogação de objectos - 1*

- O acesso eficiente aos diferentes ficheiros existentes num sistema de ficheiros impõe uma organização hierarquizada da informação visando a determinação tão expedita quanto possível, a partir do *nome*, dos atributos de um ficheiro e da localização do seu conteúdo.
- Este objetivo é concretizado, seguindo as regras da arquivística tradicional, através do recurso a *diretórios* (também designados de *pastas* na perspetiva puramente arquivística) e à formação de uma hierarquia de *diretórios*.
- Como já foi referido, um *diretório* é concebido com um *array* de *entradas de diretório* de dimensão variável, cada entrada contendo o *nome* e os *atributos*, ou uma referência à localização do *atributos*.

DETI

# *Catalogação de objectos - 2*

- Quando um directório é criado, o seu conteúdo é formado pelo conjunto de entradas que podem ser armazenadas num bloco da zona de dados. Destas, só as duas primeiras são ocupadas: a primeira com uma referência ao próprio diretório e a segunda com uma referência ao diretório imediatamente acima na hierarquia.
- À medida que outros ficheiros vão sendo criados e incluídos no diretório, as restantes entradas vão sendo ocupadas e, quando não houver já entradas livres, um novo bloco será associado ao conteúdo informativo do directório para aumentar o tamanho do *array*.




*conteúdo de um directório entendido como um array de entradas de directório de comprimento variável*

DETI

# *Catalogação de objectos - 3*

- O acesso a um ficheiro elencado num dado diretório supõe a pesquisa do *array* de *entradas de diretório* até que a entrada referente a esse ficheiro seja localizada. Se o número de ficheiros listados não for muito elevado (estiver limitado a algumas dezenas, e esse é geralmente o caso), a realização de uma pesquisa linear do *array* é geralmente considerada aceitável.
- O problema coloca-se então ao nível do número de elementos do *array* que podem ser armazenados num bloco da zona de dados. Quanto maior for esse número, menor será o número de acessos a blocos do dispositivo. O que significa naturalmente reduzir o tamanho dos dois campos constituintes.
- Reduzir o campo dos *atributos* implica no fundo garantir um armazenamento tão compacto quanto possível, se ele for efetuado diretamente na própria entrada de diretório, ou alternativamente efetuá-lo fora da entrada, numa estrutura de dados própria, e manter na entrada apenas uma referência à sua localização.

DETI

# *Catalogação de objectos - 4*

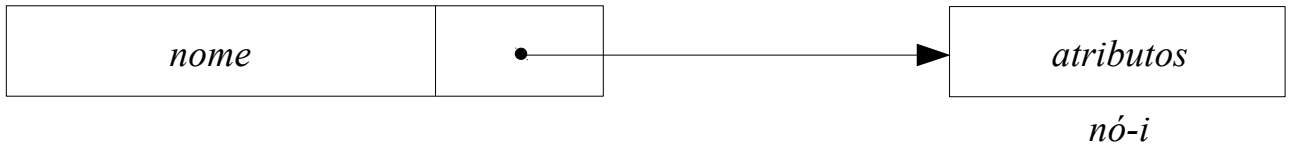
## Atributos localizados na entrada de directório

| *nome* | *atributos* |
|---|---|

- Quando a gestão de espaço supõe um armazenamento disperso
  - A descrição do conteúdo informativo do ficheiro é efetuada usando uma lista ligada em que os nós são os próprios blocos de armazenamento, ou existe uma lista de referência única comum a todo o sistema de ficheiros (FAT)
  - Só desta maneira é possível restringir a descrição a apenas dois campos: *início* e *tamanho*.
  - A existência de múltiplas listas de referência exigiria o seu armazenamento separado, ou uma limitação drástica ao tamanho máximo que um ficheiro poderia assumir.

# *Catalogação de objectos - 5*

## Atributos localizados fora da entrada de directório




- Neste caso, o tamanho do campo dos *atributos* é reduzido ao mínimo, sendo aí tão só armazenado uma referência para uma estrutura de dados que efetivamente contém os *atributos*, o chamado *nó-i*. Esta referência é tipicamente a entrada de uma tabela de nós-i (*array* cujos elementos são nós-i) e funciona como o *identificador interno* do ficheiro.
- O maior inconveniente desta abordagem é impor um número máximo de ficheiros que podem ser armazenados no sistema de ficheiros, já que se estabelece uma correspondência biunívoca entre cada ficheiro e o nó-i onde são descritos os seus atributos.

DETI

# *Catalogação de objectos - 6*

- A redução do campo *nome* da entrada de diretório coloca um problema mais complicado. Impor um número pequeno de caracteres para o armazenamento do *nome* não é hoje em dia uma opção considerada aceitável. A tendência atual é permitir a utilização de identificadores de comprimento mais ou menos longo, formados eventualmente por uma sucessão de palavras.
- Uma solução possível é ainda assim limitar o tamanho do campo a um número de caracteres assumido como razoável: 64 ou 256, por exemplo. Esta alternativa permite que os algoritmos de pesquisa não sejam muito complexos e, por isso, em princípio eficientes e baseia-se na suposição, que é verificada na prática, que quando os nomes são escolhidos por seres humanos, nunca serão demasiado longos.

# *Catalogação de objectos - 7*

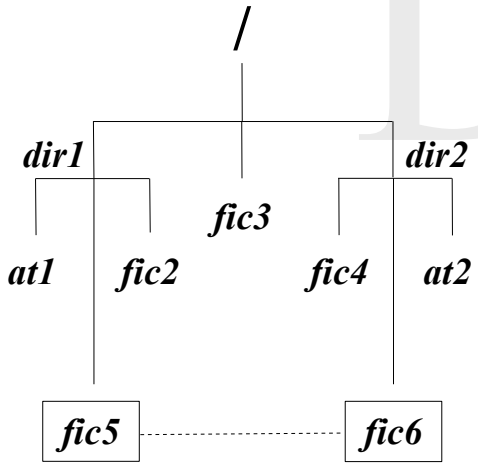
- Por outro lado, se não se quiser impor um limite efetivo ao tamanho dos identificadores que nomeiam os ficheiros, pode sempre seguir-se uma das alternativas seguintes
  - fixar o tamanho do campo *nome* a um valor pré-definido e usar, sempre que se justifique, entradas de directório sucessivas para armazenar o identificador do ficheiro
  - organizar o conteúdo informativo do diretório em duas partes: um *array* de entradas de diretório e uma região para armazenamento dos identificadores dos ficheiros de uma forma compacta; desta forma, a contribuição do campo *nome* para o tamanho da entrada de diretório seria mínima.

DETI

# *Catalogação de objectos - 8*

- Quando os atributos dos ficheiros são armazenados em estruturas de dados separadas, *nós-i*, a partilha de ficheiros no mesmo sistema de ficheiros pode efectuar-se de um modo muito mais eficiente do que recorrendo a *atalhos*.

DETI




- Neste tipo de organização, a hierarquia de directórios deixa de constituir uma *árvore* e passa a ser um *gráfico acíclico dirigido*.

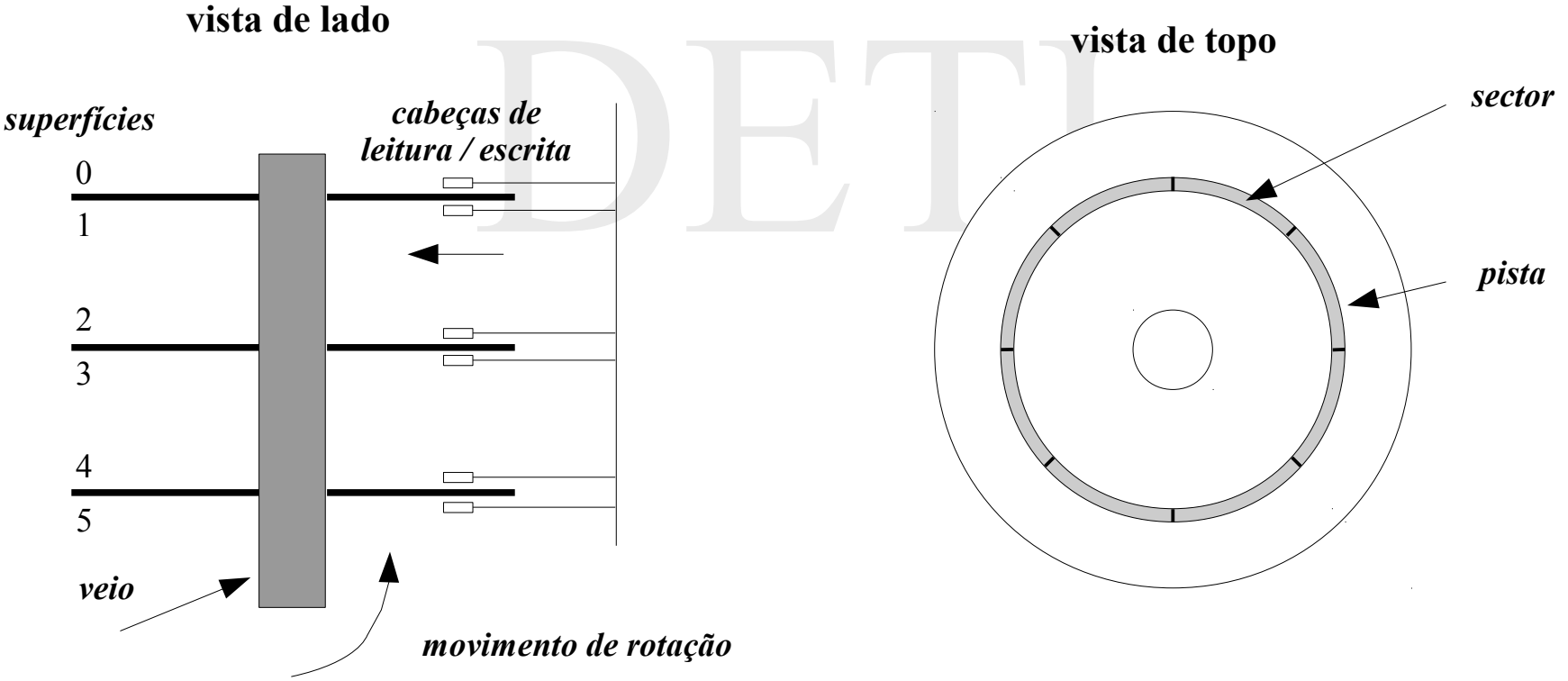
# *Catalogação de objectos - 9*

- Como a figura ilustra, a nomeação do ficheiro em cada um dos diretórios onde ele está elencado, pode ser diferente e, portanto, o *nome* do ficheiro passa a ser uma propriedade meramente local [aos diretórios de referência] e não uma propriedade central do objeto (esta será o seu *identificador interno* – número do nó-i onde os seus atributos estão armazenados).
- Assim, os *atributos* do ficheiro têm que conter um elemento adicional que conta o número de referências existentes no sistema de ficheiros (entradas de diretório) associadas com esse ficheiro.
- O *apagamento* de um ficheiro num diretório passa a representar o apagamento da entrada de diretório correspondente e não necessariamente a remoção do ficheiro do sistema de ficheiros. Isto só ocorrerá quando o *contador de referências* assumir o valor zero.
- Por essa razão, cada entrada de diretório constitui aquilo que se designa de *ligação profunda* (*hard link*) ao ficheiro, por oposição a um *atalho* que constitui uma *ligação simbólica* ao ficheiro que referencia.

DETI

# *Disco magnético - 1*

- As unidades de disco magnético, ou de *disco duro*, como comummente são designadas, são os dispositivos físicos mais usados no armazenamento *on-line* de informação. As principais razões que o justificam, têm a ver com a grande capacidade de armazenamento, a fiabilidade e o baixo preço por bit.




DETI

# *Disco magnético - 2*

- Um disco magnético é formado por um pilha de discos de alumínio, ou de um material cerâmico, em cujas superfícies é depositado uma película de um material ferromagnético.
- A pilha é sustentada por um veio central que a faz girar a uma velocidade típica entre as 3800 e as 7200 rotações por minuto (60 a 120 rot/s). Um dispositivo conversor, designado *cabeça de leitura-escrita*, que transforma bidirecionalmente fluxos magnéticos em correntes elétricas, desloca-se radial e solidariamente sobre cada uma das superfícies. Torna-se, assim, possível a deteção e o estabelecimento de orientações bem-definidas de magnetização sobre regiões muito pequenas.
- A coroa circular, descrita por uma cabeça de leitura-escrita no movimento de rotação sobre uma superfície, é chamada *pista*. Cada *pista* é, por sua vez, dividida para efeitos de armazenamento de informação em múltiplos *setores* que correspondem aos elementos do *array* de blocos da abstração usada para descrever o dispositivo. O conjunto de todas as pistas correspondentes ao mesmo deslocamento radial das cabeças é designado de *cilindro*.

# *Disco magnético - 3*

- Dados os movimentos mecânicos envolvidos no posicionamento de uma cabeça de leitura-escrita sobre um setor particular, dois tempos têm que ser tidos em conta na formação do *tempo de posicionamento* (*positioning time*). São eles
  - *tempo de pesquisa* (*seek time*) – representa o tempo médio necessário ao posicionamento da cabeça sobre a pista onde está localizado o setor; existem milhares de pistas nos dispositivos atuais (o tempo de deslocamento da cabeça entre o centro e a periferia da superfície está hoje em dia na gama dos 5 a 12 ms)
  - *tempo de latência* (*latency time*) – representa o tempo médio necessário, uma vez posicionada a cabeça, a que o sector pretendido passe imediatamente de baixo da cabeça; existem centenas de setores em cada pista nos dispositivos atuais (o tempo de latência está na gama dos 4 a 8 ms para as velocidades de rotação referidas atrás).

DEI

# *Disco magnético - 4*

- Uma vez posicionada a cabeça sobre o setor, a leitura ou a escrita de informação processa-se bit a bit por transferência entre a memória interna do controlador e o dispositivo. O *ritmo de transferência* depende diretamente da velocidade de rotação.
- Para que o *ritmo de transferência* possa ser sustentado, é necessário ler ou escrever não um único sector, mas setores logicamente sucessivos. O *ritmo de transferência* pode ser aumentado tirando partido das diferentes superfícies existentes, com leitura ou escrita em simultâneo dos setores pertencentes ao mesmo cilindro. Deste modo, pode atingir-se *ritmos de transferência* na ordem de grandeza das centenas de Mbytes/s.
- Qualquer que seja o caso, porém, há sempre um *overhead* de cerca de uma dezena de ms no acesso ao disco magnético no início de cada nova transferência.
-

DEI

# *Cache do disco - 1*

- Para minimizar este *overhead* de acesso, o sistema de operação atua de duas maneiras diferentes
  - *escalonamento do dispositivo* – ordenação dos pedidos de acesso de modo a minimizar o tempo de pesquisa
  - *cache de blocos do dispositivo* – criar em memória principal uma região para armazenamento temporário de cópias de setores do dispositivo mais frequentemente acedidos para garantir um tempo de acesso da mesma ordem de grandeza da transferência de informação entre regiões de memória distintas.

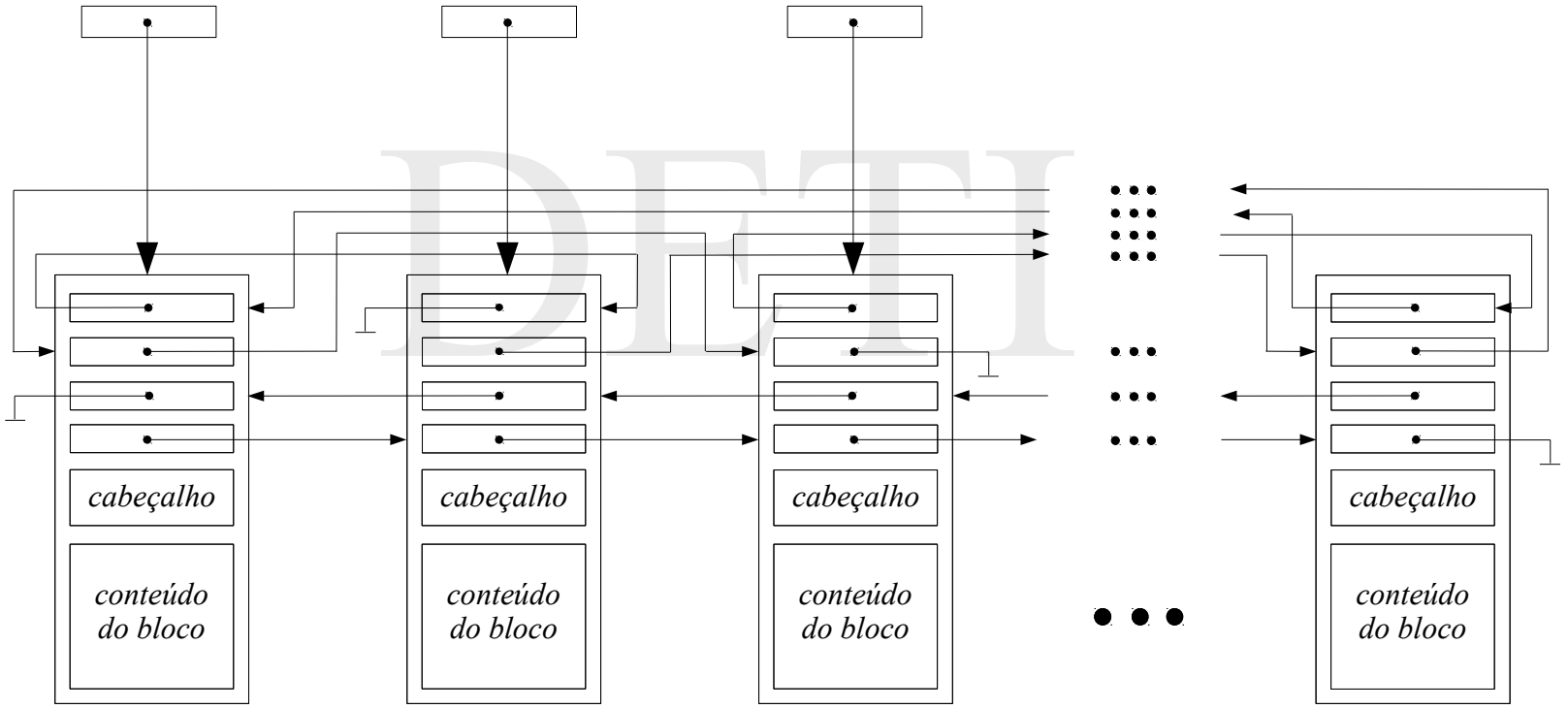
DETI

# *Cache do disco - 2*

***ponteiro para o início da lista biligada baseada no número do bloco***

***ponteiro para o início da lista biligada baseada no tempo do último acesso***

***ponteiro para o fim da lista biligada baseada no tempo do último acesso***




DETI

# *Cache do disco - 3*

- Algoritmo de leitura de um bloco
  - Os nós da *cache* são pesquisados para verificar se o bloco já lá se encontra
    - caso não se encontre
      - um novo nó é selecionado para albergar o bloco e simalizado com o seu nº
      - o bloco é lido do dispositivo e armazenado no nó
      - o *status* é posto a *mesmo*
    - o conteúdo é copiado para o *buffer* cujo endereço é passado
- Algoritmo de escrita de um bloco
  - Os nós da cache são pesquisados para verificar se o bloco já lá se encontra
    - caso não se encontre
      - um novo nó é selecionado para albergar o bloco e simalizado com o seu nº
    - o conteúdo do *buffer*, cujo endereço é passado, é copiado para lá
      - o *status* é posto a *altetado*

DETI

# *Cache do disco - 4*

- porque o número de nós da *cache* é finito, sempre que não haja nós livres na *cache*, o nó que não é acedido há mais tempo, é selecionado para substituição: se o seu *status* for igual a *alterado*, o conteúdo é primeiro transferido para o dispositivo, antes que o nó seja disponibilizado para uma nova atribuição.

DEI

# *Leituras sugeridas - 1*

*Operating Systems Concepts,* Silberschatz, Galvin, Gagne, John Wiley & Sons, 8th Ed

Capítulo 10: *File System*

Conceito de ficheiro

Capítulo 11: *Implementing File Systems* (Secções 11.1 a 11.5 e 11.9)

Implementação de um sistema de ficheiros

*Capítulo* 12*: Secondary storage structure* (Secção 12.1.1)

Disco magnético

*Modern Operating Systems,* Tanenbaum, Prentice-Hall International Editions, 3rd Ed

Capítulo 4: *File Systems* (Secções 4.1 a 4.3.4 e 4.5)

Memória de massa

Conceito de ficheiro

Implementação de um sistema de ficheiros

Exemplos

DETI

# *Leituras sugeridas - 2*

*Operating Systems, W. Stallings,* Prentice-Hall International Editions, 7th Ed

- Capítulo 12: *File management* (Secções 12.1 a 12.7 e 12.9 a 12.11)
  - Conceito de ficheiro
  - Implementação de um sistema de ficheiros
  - Exemplos
- Capítulo 11: *I/O management and disk scheduling* (Secções 11.5 e 11.7)
  - Disco magnético
  - Cache de disco

DETI